A Time Petri Net Description of Membrane Systems with Priorities, Dissolution, and Promoters/Inhibitors

Péter Battyányi, György Vaszil*

Department of Computer Science, Faculty of Informatics
University of Debrecen
Kassai út 26, 4028 Debrecen, Hungary
{battyanyi.peter,vaszil.gyorgy}@inf.unideb.hu

Summary. We continue the investigations of the connection between membrane systems and time Petri nets by extending simple symbol-object membrane systems with promoters/inhibitors, membrane dissolution and priority for rules. By constructing the simulating time Petri net, we retain one of the main characteristics of the Petri net model, namely, the firings of the transitions can take place in any order, there is no need to introduce maximal parallelism in the Petri net semantics. Instead, we substantially exploit the gain in computational strength obtained by the introduction of the timing feature for Petri nets.

1 Introduction

Several models have emerged in the past decades to model distributed systems with interactive, parallel components. One of them was developed by C.A. Petri [6], and since then the Petri nets have become the underlying system of a vast field of research with a considerable practical interest on the other hand. The theory of membrane systems was invented by Gh. Păun [4], and it has proved to be a very convenient and many-sided model of distributed systems with concurrent processes. Here we continue the investigations concerning the relationship of these two computational models.

Place/transition Petri nets are bipartite graphs, the conditions of the events of a distributed system are represented by places and directed arcs connect the places to the transitions, that model the events. The conditions for the events are expressed by tokens, an event can take place, i.e., a transition can fire, if there are enough tokens in the places at the ends of the incoming arcs of a transition. These places are called preconditions. The outgoing edges of a transition represent the

* Gy. Vaszil was supported by grant K 120558 of the National Research, Development and Innovation Office of Hungary (NKFIH), financed under the K 16 funding scheme.

postcondition of the events. Firing of a transition means removing tokens from the preconditions and adding them to the postconditions. The number of tokens moved in this way are prescribed by the multiplicities of the incoming and outgoing arcs.

In some cases the original place/transition model has turned out not to be satisfactory, for example, we are not able to model systems where a certain order of events must be taken into account. In order to deal with this difficulty, various extensions of the Petri net model have appeared. In this paper we deal with the time Petri net model developed by P.M. Merlin [3]. In this model, time intervals are associated to transitions. The local time observed from a transition can be modified by the Petri net state transition rules, and a transition can fire only if its observed time lies in the interval assigned to the transition by the construction of the model. In this way, the computational power of Petri nets is increased: the time Petri net model is Turing complete in contrast with the original state/transition Petri net.

Membrane systems are models of distributed, synchronized computational systems ordered in a tree-like structure. The building blocks are compartments, which contain multisets of objects. The multisets evolve in each compartment in a parallel manner, and the compartments, in each computational step, wait for the others to finish their computation, hence the system acts in a synchronized manner. In every computational step, the multisets in the compartments evolve in a maximal parallel manner, this means that, in each step, as many rules of the compartment are applied simultaneously as possible.

In this paper we continue the research on the connection between time Petri nets and membrane systems initiated in [1]. We extend the basic construction of the time Petri net simulating a symbol object membrane system developed in [1] in order to represent some more membrane computational tools like promoters/inhibitors, membrane dissolution and priority of rules. One of the main features of our construction is that the unsynchronized characteristics of Petri nets is retained when a Petri net equivalent of a membrane system is presented. That is, unlike the construction in [2] (and unlike the constructions in many other Petri net descriptions of membrane systems), we do not stipulate that the Petri nets should perform their computational steps in a maximal parallel manner, the attached time intervals provide the synchronization in the corresponding Petri nets. Similarly, there will be no need to introduce any other "special features" (beside the feature of time) to be able to capture the effect of priorities, the use of promoters/inhibitors, or membrane dissolution.

2 Membrane Systems

Membrane systems are computational models operating on multisets. A finite multiset over an alphabet O is a mapping $M : O \to \mathbb{N}$, where $\mathbb{N}$ is the set of non-negative integers. The number $M(a)$ for $a \in O$ is called the multiplicity of a in

M. We write that $M_1 \subseteq M_2$ if for all $a \in O$, $M_1(a) \leq M_2(a)$. The union or sum of two multisets over O is defined as $(M_1 + M_2)(a) = M_1(a) + M_2(a)$, while the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2)(a) = M_1(a) - M_2(a)$ for all $a \in O$. The set of all finite multisets over an alphabet O is denoted by $\mathcal{M}(O)$; the empty multiset is denoted by $\emptyset$.

The notation $\mathbb{N}_{>0}$ stands for the set of positive integers, while $\mathbb{Q}$ and $\mathbb{Q}_{\geq 0}$ denotes the set of rational numbers and non-negative rational numbers and $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ the set of real numbers and non-negative real numbers, respectively.

We define the notion of the basic symbol-object membrane system [5] together with the additional features discussed in Section 5. A membrane system (or P system) is a tree-like structure of hierarchically arranged membranes. The outermost membrane is usually called the *skin* membrane. The membranes are labeled by natural numbers $\{1, \ldots, n\}$, and we use the notation m_i for a membrane labeled by i. Each membrane, except for the *skin* membrane, has its parent membrane. We use μ for representing the structure of the membrane system itself, in fact, the structure itself can be given as a balanced string of left and right brackets indexed by their labels. For example, $\mu = [_{skin}\, [_1\, [_2]_2\, [_3]_3\, [_4]_4\,]_{skin}$. Here, the skin membrane has two submembranes, while region 1 also contains two embedded regions. Abusing the notation, $\mu(i) = k$ can also mean that the parent of the i-th region is region k.

The regions contain multisets over a finite alphabet O. The contents of the regions of a P system evolve through rules associated with the regions. The rules constitute the micro steps of the computations. They are applied in a maximal parallel manner. A computational step is the macro step of the process: it ends when each of the regions have finished their computations. A computational sequence is a sequence of computational steps.

Here we make the assumption that the computational steps in the regions consist of two phases: first the rule application phase produces from the objects on the left-hand sides of the rules the labeled objects on the right-hand sides (the labels of the labeled objects describe the way they should be moved between the regions: stay where they are, move to the parent region, or move into one of the children regions); then we have the communication phase when the labels are removed and all the objects are transported to the regions indicated by their labels. The P system gives a result when it halts, i.e., when no more rules can be applied in any of the regions. The result is a number or a tuple of natural numbers counting certain objects in the membrane designated as the output membrane.

A *P system* of degree $n \geq 1$ is $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$ where O is an alphabet of objects, μ is a membrane structure of n membranes, $w_i \in \mathcal{M}(O)$ with $1 \leq i \leq n$ are the initial contents of the n regions, R_i with $(1 \leq i \leq n)$ are the sets of evolution rules associated with the regions; they are of the form $u \to v$, where $u \in \mathcal{M}(O)$ and $v \in \mathcal{M}(O \times tar)$, and $tar = \{here, out\} \cup \{in_j \mid 1 \leq j \leq n\}$.

Unless stated otherwise, we consider the n-th membrane as the output membrane. A configuration is the sequence $W = (w_1, \ldots, w_n)$, where w_k is the multiset contained by membrane m_k $(1 \leq k \leq n)$. For a rule $r : u \to v \in R_i$, we denote

by $lhs(r)$ and $rhs(r)$ the left-hand side and the right-hand side of r, respectively (u and v, for the rule $u \to v$). By the application of a rule $u \to v \in R_i$ we mean the process of removing the elements of u from the multiset w_i and extending w_i with the labeled elements, which are called *messages*. As a result, during a computational step, a region can contain both elements of O and messages. An *intermediate configuration* is an n-tuple of multisets over $O \cup (O \times tar)$. We say that W is a *proper configuration*, if $w_i \in \mathcal{M}(O)$ for each of its regions w_i.

The communication phase means that the elements coming from the right-hand sides of the rules of region i should be added to the regions as specified by the target indicators associated with them. If the right-hand side of a rule r contains a pair $(a, here) \in O \times tar$, then $a \in O$ is added to region i, the region where the rule is applied. If it contains $(a, out) \in O \times tar$, then a is added to the parent region of region i. If it contains $(a, in_j) \in O \times tar$, then a is added to the contents of region j. In the latter case, $\mu(j) = i$ holds.

Given a (proper) configuration W, we obtain a new (proper) configuration W' by executing the two phases of the transformations determined by the maximal parallel sets of rules chosen for each compartment of the membrane system. We call this a computational step, and denote it by $W \Rightarrow W'$.

We might consider additional features being present in the membrane system. First of all, we can add promoters and inhibitors to the rules. These are multisets of objects that regulate the rule applications in a way that the promoter $z \in O^*$ assigned to the rule r prescribes that z must be present in the region where the rule is applied, while the inhibitor $\neg z$ with $z \in O^*$ prevents the rule from being applied if z is present in the region.

Second, we deal with the so-called membrane dissolution. The set of objects is extended with an additional element δ that can appear on the right-hand sides of the rules. If δ appears in a rule r which is applied in the i-th region for some i, then the communication phase is executed as before and, as the result of the presence of δ in region i, the region together with its set of rules R_i disappears from the P system. This means that the elements of region i (except δ, which disappears) are passed over to the region containing i (the parent region) and the rules in R_i are not applied anymore. Note that the outermost region (the *skin* region) cannot dissolve.

Finally, we consider a priority relation among rules. That is, we consider a partial order relation (an antisymmetric and transitive relation) ρ_i on the set R_i for each $1 \leq i \leq n$. We say that r' has priority over r, or r' has higher priority than r, if $(r', r) \in \rho_i$. In this case, if both r' and r were applicable in a maximal parallel step, then r is suppressed, that is, not allowed to be applied.

In Section 5 we show that all these features can be smoothly modeled by time Petri nets. The advantage of using time Petri nets instead of the Petri net models mostly applied in the literature (see e.g. [2]) is the fact that the usual order of the firings of the transitions is preserved: we do not inflict any additional firing condition on the transitions of the Petri nets (like the requirement that the

transitions fired in a computational step should constitute a maximal multiset of fireable transitions).

3 Time Petri Nets

In this section, following the definitions in [7] we define time Petri nets- a model rendering time intervals to transitions along the concept of Merlin [3]. First of all we define the underlying place/transition Petri nets, and then extend this model to the timed version.

A *Petri net* is a tuple $U = (P, T, F, V, m_0)$ such that

1. P, T, F are finite, where $P \cap T = \emptyset$, $P \cup T \neq \emptyset$ and $F \subseteq (P \times T) \cup (T \times P)$,
2. $V : F \to \mathbb{N}_{>0}$,
3. $m_0 : P \to \mathbb{N}$.

The elements of P and T are called *places* and *transitions*, respectively. The elements of T are the *arcs*, and F is the *flow relation* of U. The function V is the multiplicity (weight) of the arcs, and m_0 is the initial marking. In general, a marking is a function $m : P \to \mathbb{N}$. We may occasionally omit the initial marking and simply refer to a Petri net as the tuple $U = (P, T, F, V)$. We stipulate that for every transition $t \in T$, there is a place $p \in P$ such that $f = (p, t) \in F$ and $V(f) \neq 0$.

Let $x \in P \cup T$. The pre- and post-sets of x, denoted by ${}^\bullet x$ and $x^\bullet$ respectively, are defined as ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$.

For each transition $t \in T$, we define two markings, $t^-, t^+ : P \to \mathbb{N}$ as follows:

$$t^-(p) = \begin{cases} V(p,t), & \text{if } (p,t) \in F, \\ 0 & \text{otherwise}, \end{cases} \quad t^+(p) = \begin{cases} V(t,p), & \text{if } (t,p) \in F, \\ 0 & \text{otherwise}. \end{cases}$$

A transition $t \in T$ is said to be enabled if $t^-(p) \leq m(p)$ for all $p \in {}^\bullet t$.

Applying the notation $\triangle\ t(p) = t^+(p) - t^-(p)$ for $p \in P$, we define the *firing* of a Petri net. Let $U = (P, T, F, V, m_0)$ be a Petri net, and m be a marking in U. A transition $t \in T$ can fire in m (notation: $m \longrightarrow^t$) if t is enabled in m. After the firing of t, the Petri net obtains the new marking $m' : P \to \mathbb{N}$ with $m'(p) = m(p) + \triangle\ t(p)$ for all $p \in P$. Notation: $m \longrightarrow^t m'$.

We obtain time Petri nets if we add time assigned to transitions of the Petri net. Intuitively, the time associated with a transition denote the last time when the transition was fired. We are considering only bounded time intervals. We present the definitions from [7], see also [8] for more information.

Definition 1. ([7]) *A time Petri net is a 6-tuple $N = (P, T, F, V, m_0, I)$ such that*

1. *the skeleton of N given by $S(N) = (P, T, F, V, m_0)$ is a Petri net, and*
2. *$I : T \to \mathbb{Q} \times \mathbb{Q}$ is a function assigning a rational interval to each transition, that is, for each $t \in T$ and $I(t) = (I(t)_1, I(t)_2)$ we have that $0 \leq I(t)_1 \leq I(t)_2$.*

We call $I(t)_1$ and $I(t)_2$ the earliest and the latest firing times belonging to t, and denote them by $eft(t)$ and $lft(t)$, respectively.

Given a time Petri nets $N = (P, T, F, V, m_0, I)$, a function $m : P \to \mathbb{N}$ is called a p-marking of N. Note that talking about a p-marking of N is the same as talking about a marking of $S(N)$.

Let $N = (P, T, F, V, m_o, I)$ be a time Petri net, $m : P \to \mathbb{N}$ a p-marking in N, and h be a function called a transition marking (or t-marking) in N, $h : T \to \mathbb{R}_{\geq 0} \cup \{\#\}$. A state in N is a pair $u = (m, h)$ such that the two markings m and h satisfy the following properties: for all $t \in T$,

1. if t is not enabled in m (that is, if $t^-(p) > m(p)$ for some $p \in {}^\bullet t$), then $h(t) = \#$,
2. if t is enabled in m (that is, if $t^-(p) \leq m(p)$ for all $p \in {}^\bullet t$), then $h(t) \in \mathbb{R}$ with $h(t) \leq lft(t)$).

The initial state is the pair $u_0 = (m_0, h_0)$, where m_0 is the initial marking and for all $t \in T$,

$$h_0(t) = \begin{cases} 0, & \text{if } t^-(p) \leq m_0(p) \text{ for all } p \in {}^\bullet t, \\ \#, & \text{otherwise .} \end{cases}$$

A transition $t \in T$ is ready to fire in state $u = (m, h)$ (denoted by $u \longrightarrow^t$) if t is enabled and $eft(t) \leq h(t)$.

We define the result of the firing for a transition that is ready to fire. Let $t \in T$ be a transition and $u = (m, h)$ be a state such that $u \longrightarrow^t$. Then the state u' resulting after the firing of t denoted by $u \longrightarrow^t u'$ is a new state $u' = (m', h')$, such that $m'(p) = m(p) + \triangle t(p)$ for all $p \in P$. Now, for all transitions $s \in T$, we have

$$h'(s) = \begin{cases} h(s), \text{ if } s^-(p) \leq m(p),\ s^-(p) \leq m'(p) \text{ for all } p \in {}^\bullet s, \\ 0, \quad \text{if } s^-(p) > m(p) \text{ for some } p \in {}^\bullet s, \text{ but} \\ \qquad s^-(p) \leq m'(p) \text{ for all } p \in {}^\bullet s, \\ \#, \quad \text{if } s^-(p) > m'(p) \text{ for some } p \in {}^\bullet s. \end{cases}$$

Hence, the firing of a transition changes not only the p-marking of the Petri net, but also the time values corresponding to the transitions. If a transition $s \in T$ which was enabled before the firing of t remains enabled after the firing, then the value $h(s)$ remains the same, even if s is t itself. If an $s \in T$ is newly enabled with the firing of transition t, then we set $h(s) = 0$. Finally, if s is not enabled after firing of transition t, then $h(s) = \#$.

Observe that we ensure that a rule can be chosen more than once in a maximal parallel step that we allow transitions to be fired several times in a row: if t is fired resulting in the new p-marking m' and $t^-(p) \leq m'(p)$ holds for all $p \in {}^\bullet t$, then $h(t)$ remains the same.

Besides the firing of a transition there is another possibility for a state to alter, and this is the time delay step. Let $u = (m, h)$ be a state of a time Petri net, and $\tau \in \mathbb{R}_{\geq 0}$. Then, the elapsing of time with τ is possible for the state u (denoted $u \longrightarrow^\tau$) if for all $t \in T$, $h(t) \neq \#$ we have $h(t) + \tau \leq lft(t)$. Then the state u',

namely the result of the elapsing of time by τ denoted by $u \longrightarrow^{\tau} u'$ is defined as $u' = (m', h')$, where $m = m'$ and

$$h'(t) = \begin{cases} h(t) + \tau, & \text{if } h(t) \neq \#, \\ \# & \text{otherwise.} \end{cases}$$

Note that definitions ensure that we are not able to skip a transition when it is enabled: a transition cannot be disabled by a time jump. This kind of semantics is called the strong semantics in the literature [8].

We remark that classic Petri nets can be obviously obtained by having $h(t) = [0, 0]$ for every transition, and no time delay step is ever made.

4 Connecting Petri Nets and Membrane Systems

First of all, we introduce the time Petri net model constructed in [1], which serves as our starting point in the constructions below. Our model relies on the correspondence between Petri nets and membrane systems described in [2], with the additional property that we do not require that our Petri net model should operate in a maximal parallel manner. In general, both by membrane systems and by Petri nets, a computational step can be considered as a multiset of rules or as a multiset of transitions, respectively. In the case of Petri nets, an application of a multiset of transitions is maximal parallel, if augmenting the multiset by any other transition results in a multiset of transitions that cannot be fired simultaneously in that configuration. In the case of membrane systems, maximal parallel execution means that, if we consider any membrane m_k, no rule of m_k can be added to our multiset of rules such that the remaining multiset still forms a multiset of executable rules in m_k. In our construction, the fireable transitions of the simulating Petri nets can be executed in any order, we do not impose a restriction on the computational sequence of the Petri nets. This involves that we have made an essential use of the time feature, since the original place/transition Petri net model is not Turing complete, unlike the majority of the symbol object membrane systems.

We remark that, similarly to membrane systems, Petri nets can also be considered as computational devices, which means that, if we start from an initial configuration such that an input is represented by the tokens contained by some designated places, then, when the computation halts, the content of the output places provide the result of the computation. Depending on the construction, the result can either be a number, or a tuple. The following statement is a reformulation of Theorem 4.2 in [1]. We present the proof with details here, since the subsequent Petri nets in the next section build upon this construction.

Theorem 1. ([1]) *Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$ be a membrane system without priorities, membrane dissolution and promoters/inhibitors.*

Then there is a time Petri net $N = (P, T, F, V, m_0, I)$ such that N halts if and only if Π halts and, if they halt, then they provide the same result.

Proof. The proof is a reinterpretation of that of Theorem 1 in [1]. We elaborate the construction again in order to keep our presentation self-contained. Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$ be a membrane system and let $N = (P, T, F, V, m_0, I)$ be the corresponding Petri net. We define N so that a computational step of Π is simulated by two subnets of N. The two subnets correspond to the two computational phases of a computational step of a membrane system, namely, the rule application and the communication phases. In our Petri net model, the tokens in the places $P_0 = O \times \{1, \ldots, n\}$ stand for the objects in the various compartments, while the tokens of the places $\bar{P}_0 = \bar{O} \times \{1, \ldots, n\}$, where $\bar{O} = \{\bar{a} \mid a \in O\}$, represent the messages obtained in the course of the rule applications. Let us see the construction in detail.

- $P = P_0 \cup \bar{P}_0 \cup \{init_{app}, init_{com}, sem, enabld\}$, where $P_0 = O \times \{1, \ldots, n\}$ and $\bar{P}_0 = \bar{O} \times \{1, \ldots, n\}$. Let $m_0(p) = w_j(a)$ for every place $p = (a, j) \in P_0$.

Intuitively, the relation $|p| = t$, where $p = (a, i) \in O \times \{1, \ldots, n\}$, means that there are as many as t objects $a \in O$ in compartment m_i. In other words, $w_i(a) = t$. On the other hand, the equality $|\bar{p}| = s$, where $\bar{p} = (\bar{b}, j) \in \bar{O} \times \{1, \ldots, n\}$, expresses the fact that there are s copies of object b that will enter into membrane m_j at the end of the computational step. The places $init_{com}, init_{app}, sem, enabld$ are places enabling the synchronization of the Petri net model.

- $T = T_0 \cup T_0^* \cup T^{\#} \cup \{t_{app}, t_{com}, t^1_{sem}, t^2_{sem}\}$,

where the transitions are defined as follows.
Let $r_{i,j} \in R_i$, where $1 \leq j \leq |R_i|$ and $1 \leq i \leq n$, be a rule in m_i. Then the transition $t_{i,j} \in T_0$ corresponds to $r_{i,j} \in R_i$. Let us define the arcs associated with $t_{i,j}$ for a fixed i and j together with their multiplicities.

- Assume $t_{i,j} \in T_0$, where $1 \leq i \leq n$ and $1 \leq j \leq |R_i|$. Then $p = (a, i) \in {}^\bullet t_{i,j}$ if and only if $a \in lhs(r_{i,j})$, and $\bar{p} = (\bar{b}, k) \in t^\bullet_{i,j}$ if and only if either $(b, in_k) \in rhs(r_{i,j})$, that is, m_i is the parent region of m_k, or $(b, out) \in rhs(r_{i,j})$, where region k is the parent region of i, or $k = j$ and $(b, here) \in rhs(r_{i,j})$.
  In addition, $enabld \in {}^\bullet t_{i,j} \cap t^\bullet_{i,j}$ $(1 \leq i \leq n, 1 \leq j \leq |R_i|)$.
  Regarding the weights of the arcs, let $p = (a, i)$ and $f = (p, t_{i,j}) \in F$. Then the weight of f is the multiplicity of $a \in O$ on the left-hand side of $r_{i,j}$, namely, $V(f) = lhs(r_{i,j})(a)$; furthermore, for $\bar{p} = (\bar{b}, k)$ and $f = (t_{i,j}, \bar{p}) \in F$, the weight of f is $V(f) = rhs(r_{i,j})(b, in_k)$ if region k is a child region of i, $V(f) = rhs(r_{i,j})(b, out)$ if region k is the parent region of i, or $V(f) = rhs(r_{i,j})(b, here)$ for $k = j$. Additionally, if $f = (t_{i,j}, enabld)$, then $V(f) = 1$ $(1 \leq i \leq n, 1 \leq j \leq |R_i|)$.

The transitions in $T^{\#}$ are in charge with the correct simulation of a maximal parallel step: they fire only if there are any enabled rules in any of the regions. Their inputs are the same as those of the elements of T_0, only their outputs differ, since they should not give rise to a change in the original distribution of the tokens before the computational step takes place.

- Let $r_{i,j} \in R_i$, where $1 \leq j \leq |R_i|,\ 1 \leq i \leq n$; then $t^{\#}_{i,j} \in T^{\#}$ is the transition checking the applicability of $r_{i,j}$. Let $p = (a,i) \in P_0$ and let $t^{\#}_{i,j} \in T^{\#}$; then $p \in {}^{\bullet}(t^{\#}_{i,j})$ and $init_{app} \in {}^{\bullet}t^{\#}_{i,j}$ and $enabld \in (t^{\#}_{i,j})^{\bullet}$ and $p \in (t^{\#}_{i,j})^{\bullet}$. In words, for a fixed i and j, $t^{\#}_{i,j}$ expects as many tokens from its outgoing places as the number of distinct objects that is necessitated by an execution of the rule $r_{i,j}$. In the meantime, a token arrives in *enabld*, which ensures the continuation of the simulation of the rule application phase. Then $t^{\#}_{i,j}$ gives back the tokens to the places in P_0.
  As regards the multiplicities, if $f = (init_{app}, t^{\#}_{i,j})$ then $V(f) = 1$, and if $f = (t^{\#}_{i,j}, enabld)$ then $V(f) = 1$; furthermore, if $f = (p, t^{\#}_{i,j})$ or $f = (t^{\#}_{i,j}, p)$ where $p = (a,i)$, then $V(f) = lhs(r_{i,j})(a)$.

The transitions in T^*_0 ensure that tokens can flow back from $\bar{P}_0$ to P_0, thus representing the communication phase of the membrane computation.

- $T^*_0 = \{s_{a,i} \mid a \in O,\ 1 \leq i \leq n\}$. Let $\bar{p} = (\bar{a}, i) \in \bar{P}_0$, then $\bar{p} \in {}^{\bullet}s_{a,i}$ and $init_{com} \in {}^{\bullet}s_{a,i}$. Moreover, if $p = (a,i)$, then $p \in s^{\bullet}_{a,i}$ and $init_{com} \in s^{\bullet}_{a,i}$. Regarding the multiplicities, each arc has multiplicity 1.

The intervals belonging to the elements of $T = T_0 \cup T^*_0 \cup T^{\#}$ are $[0,0]$. The rest of the transitions are defined as follows.

- t_{app} connects *enabld*, and hence the rule application part of the Petri net with the semaphore: $enabld \in {}^{\bullet}t_{app}$ and $sem \in t^{\bullet}_{app}$. Moreover, $V(enabld, t_{app}) = 1$ and $V(t_{app}, sem) = 2$ and $I(t_{app}) = [1,1]$.

The role of t_{app} is to guarantee that a sequence of firings of transitions correctly simulates a maximal parallel application of membrane rules: every transition, $t_{i,j}$ $(1 \leq j \leq |R_i|,\ 1 \leq i \leq n)$, corresponding to a rule execution can fire only if a token is found in *enabld*. On the other hand, if no transition $t_{i,j}$ can fire, then the transition t_{app} connected only to *enabld* will be activated after a time unit's delay.

- t_{com} connects $init_{com}$, and the communication part of the Petri net with the semaphore: $init_{com} \in {}^{\bullet}t_{com}$ and $sem \in t^{\bullet}_{com}$. Moreover, $V(init_{com}, t_{com}) = V(t_{com}, sem) = 1$ and $I(t_{com}) = [1,1]$.
  The role of the semaphore is to make sure that the simulation of the rule application and the communication phases takes place in an alternating order. This is achieved by the following machinery.
- Let t^1_{sem} and t^2_{sem} be transitions of the semaphore. Then $sem \in {}^{\bullet}(t^1_{sem})$ and $sem \in {}^{\bullet}(t^2_{sem})$; furthermore, $init_{app} \in (t^1_{sem})^{\bullet}$ and $init_{com} \in (t^1_{sem})^{\bullet}$. If $f = (sem, t^2_{sem})$, then $V(f) = 2$. The weights of the other arcs are 1. In addition, $I(t^1_{sem}) = [1,1]$ and $I(t^2_{sem}) = [0,0]$.

To sum up the above construction: a computational step of a membrane system is split into a rule application and a communication phase, and those two phases are simulated separately and in an alternating order. The simulation of a phase finishes when no more rule applications are possible, hence we ensure that a maximal

parallel step is correctly simulated. When the rule application phase finishes its operation, 2 tokens are sent to the semaphore via t_{app}, and the simulation of the communication phase can immediately begin by forwarding the 2 tokens to $init_{com}$. Otherwise, when the communication phase finishes its operation, only 1 token is sent to the semaphore, so the rule application phase is initiated after a time unit's wait. The structure of the various subnets are described in Figures 1, 2 and 3, respectively.

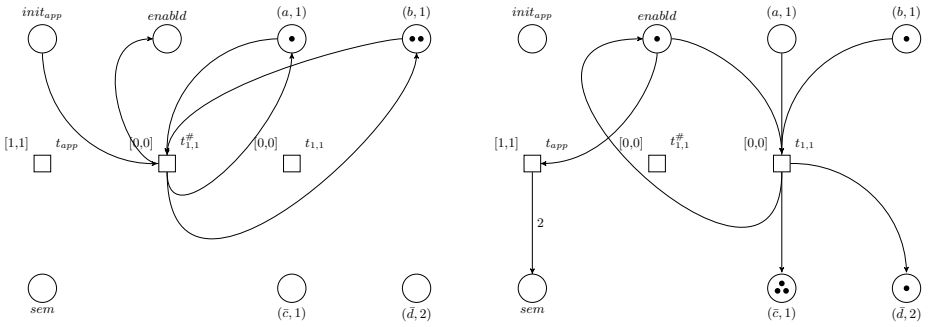

Fig. 1. Assume $a, b^2 \in w(1)$ and $r_{1,1} = ab \to c^3(d, in_2)$, where m_2 is child of m_1. The figure shows the result of a single application of the rule in a split table: to the left is the subnet testing the applicability of $r_{1,1}$ and to the right is the application of the rule itself. The rule consumes an a and a b in region 1 and three tokens are sent to the place $(\bar{c}, 1)$, and one token to $(\bar{d}, 2)$, in accordance with the fact that three objects of c should be added to region 1, and one copy of d should be added to region 2 in the communication phase.

By this, we have simulated a membrane system with a time Petri net such that in the Petri net model no restriction on the transitions is made: the transitions that are ready to fire can be fired in any order. □

5 Extending the Correspondence to Membrane Systems with More Features

In this section we examine the possibility of extending our core model to Petri nets that are able to represent various properties of membrane systems, such as the presence of promoters/ inhibitors, membrane dissolution and priority among rules. The obtained Petri nets each build upon the basic model defined in the previous section, so, in most of the cases, we restrict ourselves to emphasize only the new elements of the constructions by which the basic Petri net model is extended.

First we begin with discussing the case of promoters and inhibitors in the membrane system. Below we present a formal definition of promoters/inhibitors

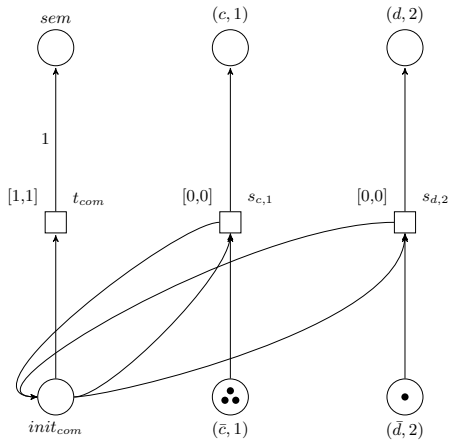

Fig. 2. The Petri net simulating the communication phase of a membrane computational step. When the simulation of a maximal parallel rule application step is finished, a token is given to the semaphore *sem*. The transitions $s_{c,1}, s_{d,2} \in T_0^*$ ensure the correct placement of the tokens corresponding to the messages.

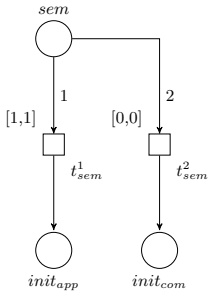

Fig. 3. The semaphore for the Petri net. When the simulation of the rule application phase of a computational step of the membrane system is complete, two tokens appear at *sem*, and then sent to $init_{com}$, activating the simulation of the communication phase of the computational step. When the simulation of the communication phase is completed, one token appears at *sem*, which is then sent to $init_0$, activating the simulation of the rule application phase of a subsequent computational step.

in a form which is unusual, but technically well suitable for the presentation of the corresponding time Petri net model.

Definition 2. *Let* $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{P})$ *be a membrane system with promoters/inhibitors, where* $\mathcal{P}(r_{j,i}) \subseteq O^* \times O^*$*, for every* $r_{j,i} \in R_i$*. Then* $\mathcal{P}(r_{j,i}) = (\rho_{j,i}, \tau_{j,i}) \subseteq O^* \times O^*$ *is a promoter/inhibitor pair for* r_j*. We denote the*

pair $(\rho_{j,i}, \tau_{j,i})$ *by* $(prom^r, inhib^r)$. *Let* $\mathcal{R}$ *be a multiset of rules. A multiset* $\mathcal{R}$ *is applicable, if each of the following conditions fulfill.*

1. $lhs(r_{j,i})(a) \cdot \mathcal{R}(j,i) \leq w_i(a)$ $(a \in O)$,
2. $prom^r(a) \leq w_i(a)$ $(r \in \mathcal{R}, a \in O)$,
3. $w_i(a) < inhib^r(a)$ $(r \in \mathcal{R}, a \in O)$.

In what follows, we give the structure of the Petri net simulating a general example of a membrane system with promoters/inhibitors.

Theorem 2. *Let* $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{P})$ *be a membrane system with promoters/inhibitors.*

Then there is a time Petri net $N = (P, T, F, V, m_0, I)$ *such that* N *halts if and only if* Π *halts, and if they halt, both of them provides the same result.*

Proof. Let Π be as in the statement of the theorem. We construct N in a way analogous to the construction of the Petri net of Theorem 1. The Petri net simulates the rule application and the communication phase separately, we only concentrate on the rule application part, since the other parts of the construction are identical to that of the proof of Theorem 1. Let us detail the proof a bit more.

- $P = P_0 \cup \bar{P}_0 \cup \{init_{app}, init_{com}, sem, enabld, contd\}$, where $P_0 = O \times \{1, \ldots, n\}$ and $\bar{P}_0 = \bar{O} \times \{1, \ldots, n\}$. Let $m_0(p) = w_j(a)$ for every place $p = (a, j) \in P_0$.

As before, if $|p| = t$, where $p = (a, i) \in O \times \{1, \ldots, n\}$, then there are as many as t objects $a \in O$ in compartment m_i. Likewise, we retain the meaning of $\bar{p} = (\bar{b}, j) \in \bar{O} \times \{1, \ldots, n\}$, where $|\bar{p}| = s$ expresses the fact that there are s copies of object b that are going to appear in membrane m_j at the end of the computational step. The places $init_{com}, init_{app}, sem, enabld, contd$ are places enabling the synchronization of the Petri net model. The new element here is the place *contd*, which is introduced in order to handle conditions 2 and 3 for rule applicability in Definition 2.

- $T = T_0 \cup T_0^* \cup T^{\#} \cup T^{\#\#} \cup \{t_{app}, t_{com}, t^1_{sem}, t^2_{sem}\}$,

where the transitions are defined as follows.

- The definitions of the transitions T_0 and T_0^* are unchanged. The construction of the arcs and their weights, with respect to T_0 and T_0^*, is exactly the same as above.

The difference lies in the definitions of $T^{\#}$ and $T^{\#\#}$. They ensure that the conditions of rule applications presented in Definition 2 are simulated correctly.

- Let $r_{i,j} \in R_i$, where $1 \leq j \leq |R_i|$, $1 \leq i \leq n$; then $t^{\#}_{i,j} \in T^{\#}$ is the transition checking conditions 1 and 2 in Definition 2. Let $p = (a, i) \in P_0$ and let $t^{\#}_{i,j} \in T^{\#}$; then $p \in {}^\bullet(t^{\#}_{i,j})$ and $init_{app} \in {}^\bullet t^{\#}_{i,j}$ and $contd \in (t^{\#}_{i,j})^\bullet$ and $p \in (t^{\#}_{i,j})^\bullet$ and $enabld \in (t^{\#}_{i,j})^\bullet$.

As regards the multiplicities, if $f = (init_{app}, t^{\#}_{i,j})$ then $V(f) = 1$, and if $f = (contd, t^{\#}_{i,j})$ or $f = (t^{\#}_{i,j}, enabld)$ then $V(f) = 1$; furthermore, if $f = (p, t^{\#}_{i,j})$ or $f = (t^{\#}_{i,j}, p)$, where $p = (a, i)$, then $V(f) = max\{lhs(r_{i,j})(a),\ prom^{r_{i,j}}(a)\}$. The time interval assigned to $t^{\#}_{i,j}$ is $[1, 1]$.

The novelty in this Petri net is the appearance of the transitions $T^{\#\#} = \{t^{\#\#}_{i,j} \mid r_{i,j} \in R_i\}$, that are responsible for the correct simulation of the inhibitors.

- Let $r_{i,j} \in R_i$, where $1 \leq j \leq |R_i|,\ 1 \leq i \leq n$; then $t^{\#\#}_{i,j} \in T^{\#\#}$ is the transition checking condition 3 in Definition 2. Let $p = (a, i) \in P_0$ and let $t^{\#\#}_{i,j} \in T^{\#}$; then $p \in {}^{\bullet}(t^{\#\#}_{i,j}) \cap (t^{\#\#}_{i,j})^{\bullet}$.
  If $f = (p, t^{\#\#}_{i,j})$ or $f = (t^{\#\#}_{i,j}, p)$, where $p = (a, i)$, then $V(f) = inhib^{r_{i,j}}(a)$.
  The time interval assigned to $t^{\#\#}_{i,j}$ is $[0, 0]$.

In words, the transitions $t^{\#\#}_{i,j}$ capture the tokens of $p = (a, i)$ in the case when $|p| \geq inhib^{r_{i,j}}(a)$. The firing sequence can continue with the simulation of the application of rule $r_{i,j}$ only if $|p| < inhib^{r_{i,j}}(a)$ holds for every $a \in O$ for which $lhs(r_{i,j})(a) > 0$.

The rest of the construction is the same as that of Theorem 1, hence we omit the details. The changes in the Petri net compared to the core model are illustrated in Figures 4. □

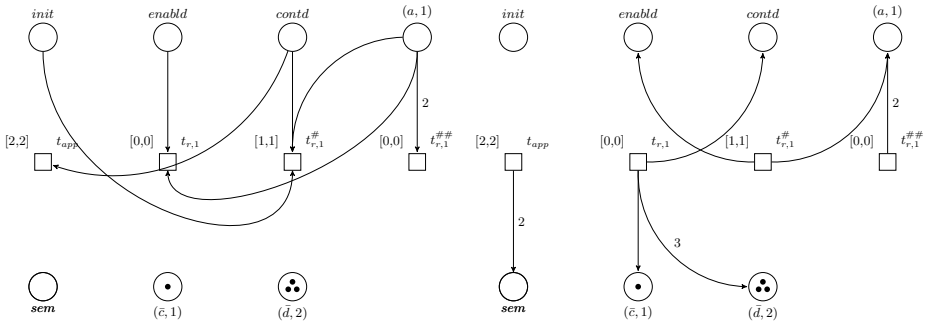

Fig. 4. The rule application phase for the Petri net, where $a \in w_1$ and $r = a \to c(d, in_2)^3$ and $prom^r(a) = 1$, $inhib^r(a) = 2$.

Next, we turn our attention to membrane systems with dissolution. Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \delta)$ be a membrane system with dissolution. We recall form our previous definitions that this means that there exists a special element $\delta \in O$, which can appear on the right side of a rule only. Assume $r \in R_i$ and $\delta \in rhs(r)$. Suppose r is chosen in the actual maximal parallel rule application of m_i. Then all the rules of R_i appearing in that computational step are executed as

usual, and, after the maximal parallel step is over, the region m_i disappears, its objects wander into the parent region and the rules R_i cease to operate. With this in mind, we construct a time Petri net simulating the operation of the membrane system in the sense below.

Theorem 3. *Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \delta)$ be a membrane system with dissolution.*

Then there is a time Petri net $N = (P, T, F, V, m_0, I)$ such that N halts if and only if Π halts, and if they halt, then both systems provide the same result.

Proof. Let Π be as in the theorem. We construct the Petri net N with the required properties. The construction again leans on the proof of Theorem 1. The rule application phase is exactly the same with one exception: places δ_i symbolizing the dissolution of membrane m_i appear. The difference manifests itself in the definition of the communication phase. Moreover, we introduce one more phase, a δ-phase, that serves for moving the elements of a previously dissolved membrane to the parent region. First of all, we define the set of places as before.

- $P = P_0 \cup \bar{P}_0 \cup \{init_{app}, init_{com}, sem, enabld, \delta_i\}$, where $P_0 = O \times \{1, \ldots, n\}$ and $\bar{P}_0 = \bar{O} \times \{1, \ldots, n\}$ and $1 \leq i \leq n$. Let $m_0(p) = w_j(a)$ for every place $p = (a, j) \in P_0$.

The only change is the presence of the places δ_i for every region m_i. Intuitively, they are indicators whether a membrane is going to disappear in the next step or has been dissolved already. This is reflected in the design of the arcs for the rule application phase. We require an extended set of transitions, since we have a third phase also that transfers the objects of the dissolved membranes to their parent membranes.

- $T = T_0 \cup T_0^* \cup T_0^\delta \cup T^\# \cup T^\flat \cup \{t_{app}, t_{com}, t_{clean}, t^1_{sem}, t^2_{sem}, t^3_{sem}\}$,

and the arcs for the rule application phase are identical to those of Theorem 1 with the only exception of the arcs pointing from $t_{i,j}$, where $r_{i,j} \in R_i$, to the place δ_i with multiplicity 1 provided $\delta \in rhs(r_{i,j})$. Thus we are only interested in the transitions T_0^*, $T^\flat$ and their corresponding arcs.

- Let $T_0^* = \{s_{a,i} \mid a \in O,\ 1 \leq i \leq n\}$ and $T_0^\delta = \{s^\delta_{a,i} \mid a \in O,\ 1 \leq i \leq n\}$. Let m_i be a region other than the skin membrane, assume m_k is its parent region. (If m_i is the skin membrane, it cannot disappear.) Let $\bar{p} = (\bar{a}, i) \in \bar{P}_0$, then $\bar{p} \in {}^\bullet s_{a,i}$ and, if $p = (a, i)$, then $p \in s^\bullet_{a,i}$. Moreover, $init_{com} \in {}^\bullet s_{a,i} \cap s^\bullet_{a,i}$. In addition, δ_i is connected with $s^\delta_{a,i}$ for every object $a \in O$, that is: $\delta_i \in {}^\bullet s^\delta_{a,k} \cap s^\delta_{a,i}{}^\bullet$ and we have $\bar{q} = (\bar{a}, k) \in s^\delta_{a,i}{}^\bullet$. Regarding the multiplicities, each arc has multiplicity 1.
  Furthermore, $I(s_{a,i}) = [1, 1]$, $I(s^\delta_{a,i}) = [0, 0]$ and $I(t_{com}) = [2, 2]$, where t_{com} is the transition connecting $init_{com}$ with the place sem.

Intuitively, if m_i is a region with parent region m_k, the communication phase transfers the tokens of $\bar{p} = (\bar{a}, i)$ to the place $p = (a, i)$, as long as the membrane

m_i exists. When m_i is marked for dissolution or has been already dissolved, that is, $|\delta_i| = 1$, then the tokens of $\bar{p} = (\bar{a}, i)$ are redirected to $\bar{q} = (\bar{a}, k)$. This is achieved by a time gap between the possible firings of the transitions $s_{a,i}$ and $s^{\delta}_{a,i}$. This implies that the elements appearing on the right hand side of the rules of a dissolved membrane find their correct place: they wander to the upper levels of the tree until they find the first ancestor region not dissolved. The main ingredients of the construction are illustrated in Figure 5.

The only missing part is the subnet directing the remaining object of a dissolved membrane into an existing container membrane. We term this phase the cleaning phase. The construction is quite simple: let m_k be a region and assume that m_i is its parent region. Then, for every place $p = (a, k)$, there corresponds a transition $t^f_p lat$ which transfers the objects of p to $q = (a, i)$ when δ_k contains a token. The place $init_{clean}$ is connected to all the transitions $t^{\flat}_p$ in order to perceive when the tidying phase is ready. After this, 3 tokens are sent to the semaphore and a new application phase activates. More formally,

- let $T^{\flat} = \{t^{\flat}_{a,k} \mid a \in O,\ 1 \leq k \leq n\}$. Assume m_i is the parent region of region m_k. Then $p = (a, k) \in {}^{\bullet}t^{\flat}_p$ and $q = (a, i) \in t^{\flat}_p{}^{\bullet}$ and $\delta_k \in {}^{\bullet}t^{\flat}_p \cap t^{\flat}_p{}^{\bullet}$. Moreover, $init_{clean} \in {}^{\bullet}t^{\flat}_p \cap t^{\flat}_p{}^{\bullet}$ and $init_{clean} \in {}^{\bullet}t_{clean}$ and $sem \in t^{\bullet}_{clean}$. Regarding the multiplicities, each arc has multiplicity 1, except for (t_{clean}, sem), which has multiplicity 3.
  Furthermore, $I(t^{\flat}_{a,i}) = [0, 0]$, $I(t_{clean}) = [1, 1]$.

This is described in Figure 6. The semaphore is extended with a new transition, t^3_{sem} which leads to the initialization of the third phase in the simulation of a maximal parallel step. The new semaphore is depicted in Figure 7. □

Finally, we tackle the problem of the representation of membrane systems with priorities in terms of Petri nets. Again, our construction is a slight modification of the core model. We introduce some pieces of information in the simulation of the rule application phase that accounts for the treatment of the priorities. Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho)$ be a membrane system with priorities. This means that $\rho \subseteq R \times R$, and the rule application is modified in the following way.

Definition 3. *Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho)$ be a membrane system with priorities. Let $r \in R_i$, if $1 \leq i \leq n$. Then r is strong-applicable, if*

1. *r is applicable, that is, $lhs(r_i) \leq w_i$, and*
2. *for every $r' \in R_i$ such that $r' > r$, r' is not applicable.*

Let $r_1,\ r_2 \in R_k$ be two rules of region m_k, assume that $(r_1, r_2) \in \rho$, that is, $r_1 > r_2$. Then, considering a computational step, r_2 can be applied if r_2 is applicable in the usual sense and, in addition, r_1 fails to be applicable in the maximal parallel step belonging to region m_k. We remark that we use priority in the strong sense: assume $w_k = a^2 b$, $r_1 = a \to c$ and $r_2 = ab \to d$. Then the result of the maximal parallel step will be ad, instead of cd, since $r_1 > r_2$ and

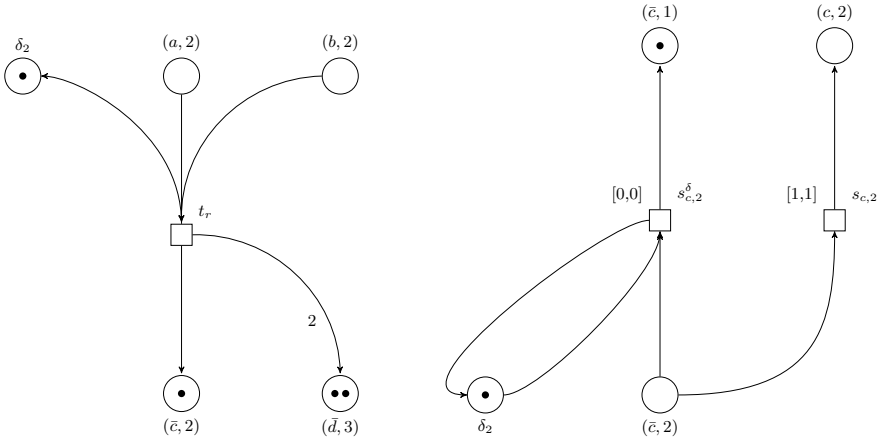

Fig. 5. The Petri net simulating a membrane system with dissolution. In this case, m_2 is dissolved, hence $s^{\delta}_{c,2}$ can be activated moving the elements of $\bar{p}(c,2)$ to $\bar{p}(c,1)$.

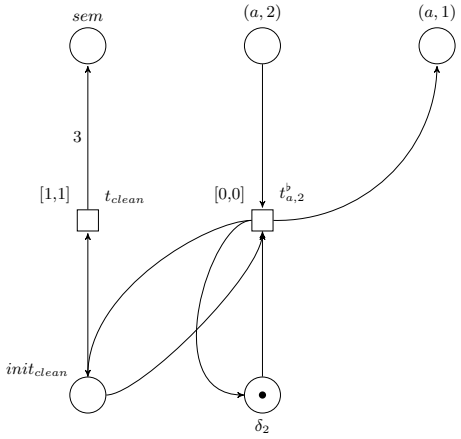

Fig. 6. The Petri net simulating the phase when the objects of a dissolved membrane are directed towards the parent membrane. Here we assume that region 1 is the parent of region 2, and the place δ_2 already has a token.

r_1 is applicable, which implies that r_2 cannot be applied in that maximal parallel step at all, even if r_1 is not applicable any more. The construction is again based on the basic construction, the only difference is that we have to pick out the applicable rules in order to obtain strong applicability. We do this by stratifying the various tasks due in the rule application phase with respect to time. Finding the strongly applicable rules takes place before the actual rule applications are simulated. Below, the place $p^A_{t_i}$ will represent the strong applicability of rule r_i.

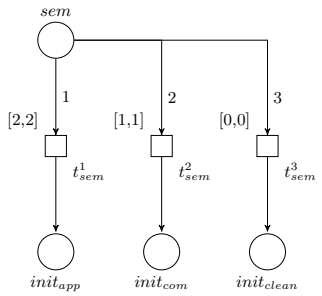

Fig. 7. The semaphore for the Petri net with dissolution. The choice of the next phase is uniquely determined by the number of tokens arriving in the place *sem*.

Now we are in a position to state the theorem on the simulation.

Theorem 4. *Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho)$ be a membrane system with priorities.*

Then there is a time Petri net $N = (P, T, F, V, m_0, I)$ such that N halts if and only if Π halts, and if they halt, then they provide the same result.

Proof. Let Π as above. We describe the Petri net N simulating Π. The only differences in comparison with the model in Theorem 1 occur by the rule application phase when we select the transitions that are candidates for the strongly applicable rules. We omit repeating the communication phase and the alternating construction of the Petri net in detail and we confine ourselves to the rule application phase only. The places are

- $P = P_0 \cup \bar{P}_0 \cup P^A \cup P^B \cup P_\rho \cup P^*_\rho \cup \{init_{app}, init_{com}, sem, enabld\}$, where $P_0 = O \times \{1, \ldots, n\}$ and $\bar{P}_0 = \bar{O} \times \{1, \ldots, n\}$ and the auxiliary places are defined as in Theorem 1. Regarding the new places, $P^A = \{p^A_{i,j} \mid r_{i,j} \in R_i, 1 \leq j \leq |R_i|\}$ and $PT^B = \{p^B_{i,j} \mid r_{i,j} \in R_i, 1 \leq j \leq |R_i|\}$. Moreover, $P_\rho = \{p_{r_i > r_j} \mid p \in P_0 \text{ and } (r_i, r_j) \in \rho\}$, and $P^*_\rho = \{p^*_{r_i > r_j} \mid p \in P_0 \text{ and } (r_i, r_j) \in \rho\}$.

The new places accomplish some bookkeeping in order to keep track of which rules are applicable and which ones are not. A token in $p^A_{i,j}$ should symbolize the applicability of an arbitrary $r_{i,j} \in R_i$, while a token in $p^B_{i,j}$ says that $r_{i,j}$ is blocked by an applicable rule of higher priority. The places P_ρ and P^*_ρ ensure that each pair r_i, r_j with $(r_i, r_j) \in \rho$ should be checked only once with the purpose of dropping out the blocked rules. We define now the new transitions together with the arcs induced by these transitions.

- $T = T_0 \cup T^*_0 \cup T^\# \cup T_\rho \cup T^*_\rho \cup T^{NA} \cup T^\flat \cup \{t_{app}, t_{com}, t^1_{sem}, t^2_{sem}\}$.

The transitions are defined as before, except for the arcs in connection with the new sets of transitions. Regarding T_ρ, where $T_\rho = \{t_{r_i > r_j} \mid r_i, r_j \in m_k,\ (r_1, r_2) \in \rho\}$ and $T^\flat = \{t^\flat_{i,j} \mid t_{i,j} \in T_0\}$. The arcs in connection with transitions T_0, T^*_0 and

$T^{\#}$, including the auxiliary transitions, are the same as in the core model. The differences emerge by the subnet checking strong applicability: if a rule $r \in R_i$ is applicable, then a token is passed over to p_t^A, where t corresponds to r. Moreover, when $r' \in R_i$ is applicable as well, and $(r', r) \in \rho$, then the token in $p_{r'>r}$ is consumed and a token is transferred to $p^*_{r'>r}$ and to p_t^B. The token in p_t^B symbolizes that r is blocked in that maximal parallel step, which is expressed at time 1, when the tokens in p_t^A and p_t^B are both consumed by the transition t^{NA}. When only tokens at $p_{t''}^A$ are left, where the corresponding rule r'' is strongly applicable, then the usual rule applications take place with the refinement that transitions in T_0 have places from p^A as incoming places as well. After finishing with the simulation of the rule applications, the tokens remaining in P^A are discarded, moreover, tokens are returned to P_ρ at time instance 3.

- Let $r_{i,j} \in R_i$, where $1 \leq j \leq |R_i|$, $1 \leq i \leq n$; then, as in the previous constructions, $t^{\#}_{i,j} \in T^{\#}$ is checking the applicability of $r_{i,j}$. Let $p = (a, i) \in P_0$ and let $t^{\#}_{i,j} \in T^{\#}$; then $p \in {}^\bullet(t^{\#}_{i,j}) \cap (t^{\#}_{i,j})^\bullet$ and $init_{app} \in {}^\bullet t^{\#}_{i,j}$ and $enabld \in {}^\bullet t^{\#}_{i,j} \cap (t^{\#}_{i,j})^\bullet$, which is a slight modification compared to Theorem 1. Enabled iw wired to each $t^{\#}_{i,j}$ with both an incoming and an outgoing arc so that the process of finding the transitions for the strongly applicable rules can continue without hindrance. Furthermore, $p^A_{i,j} \in (t^{\#}_{i,j})^\bullet$.
  Regarding the multiplicities, $V((init_{app}, t^{\#}_{i,j})) = 1$, and if $f = (t^{\#}_{i,j}, enabld)$ then $V(f) = 1$; furthermore, if $f = (p, t^{\#}_{i,j})$ or $f = (t^{\#}_{i,j}, p)$ where $p = (a, i)$, then $V(f) = lhs(r_{i,j})(a)$. in addition, the multiplicity of $(t^{\#}_{i,j}, p^A_{i,j})$ is 1.
- Now, we turn to the operations of the transitions T_ρ. Let $r_{i,j}$, $r_{i,k} \in R_i$ with $r_{i,k} > r_{i,j}$. Then $t_{r_{i,k}>r_{i,j}} \in T_\rho$, and for $p^A_{i,k}$ and $p^A_{i,j}$, both of them are in ${}^\bullet t_{r_{i,k}>r_{i,j}} \cap t^\bullet_{r_{i,k}>r_{i,j}}$. Moreover, $p_{r_{i,k}>r_{i,j}} \in {}^\bullet t_{r_{i,k}>r_{i,j}}$ and $p^*_{r_{i,k}>r_{i,j}} \in t^\bullet_{r_{i,k}>r_{i,j}}$. Furthermore, $p^B_{r_{i,j}} \in t^\bullet_{r_{i,k}>r_{i,j}}$ and, if $r_i p \in R_i$ is arbitrary, $p^A_{t_p}$ and $p^B_{t_p} \in {}^\bullet t_p^{NA}$. Finally, $\cup T^*_\rho$ return the elements of $p^*_{r_{i,k}>r_{i,j}}$ back to $p_{r_{i,k}>r_{i,j}}$, when the rule application phase is over. That's why $p^*_{r_{i,k}>r_{i,j}} \in {}^\bullet t^*_{r_{i,k}>r_{i,j}}$ and $p_{r_{i,k}>r_{i,j}} \in (t^*_{r_{i,k}>r_{i,j}})^\bullet$.
  The multiplicities of all the new arcs is 1.
  The elements of $T^\flat$ collect the tokens that might remain in the places P^A when the simulation of the maximal parallel step is over. We have $p^A_{i,j} \in {}^\bullet t^\flat_{i,j}$ for every index pair i, j such that $t^\flat_{i,j} \in T^\flat$. The multiplicities of the arc is 1.
  The elements of T_0 are extended with one more arc: for every $t \in T_0$ we have $p^A_t \in {}^\bullet t \cap t^\bullet$ with multiplicity 1.
  The timing makes sure that finding the strongly applicable elements precedes the rule applications themselves. For each $t_{r_{i,k}>r_{i,j}} \in T_\rho$, we have $I(t_{r_{i,k}>r_{i,j}}) = [0, 0]$, moreover, $I(t^{NA}) = [1, 1]$ and $I(t) = [2, 2]$, if $t \in T_0$. Finally, $I(t^\flat_{i,j}) = I(t^*_{r'>r}) = [3, 3]$.
- Regarding $t_{i,j} \in T_0$, where $1 \leq i \leq n$ and $1 \leq j \leq |R_i|$ is as before, let $p = (a, i) \in {}^\bullet t_{i,j}$ if and only if $a \in lhs(r_{i,j})$. In addition, $enabld \in {}^\bullet t_{i,j} \cap t^\bullet_{i,j}$ $(1 \leq i \leq n, 1 \leq j \leq |R_i|)$.

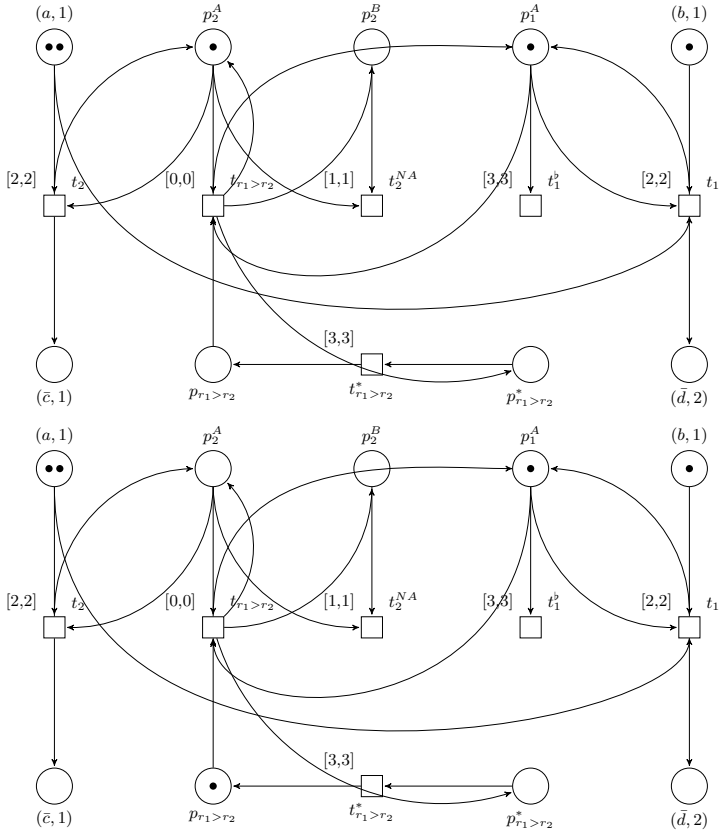

Fig. 8. Assume $w_1 = a^2b$ and r_1, $r_2 \in R_1$, $r_1 = ab \to d$, $r_2 = a \to c$ such that $r_1 > r_2$. Then only t_1 can fire: transition $t_{r_1>r_2}$ delivers a token to place p_2^B, and at time instance 1 the tokens from p_2^A and p_2^B are removed by transition t_2^{NA}.

As regards the weights of the arcs, let $p = (a, i)$ and $f = (p, t_{i,j}) \in F$. Then the weight of f is the multiplicity of $a \in O$ on the left-hand side of $r_{i,j}$, namely, $V(f) = lhs(r_{i,j})(a)$. If $f = (t_{i,j}, enabld)$ or $f = (enabld, t_{i,j})$, then $V(f) = 1$ $(1 \leq i \leq n, 1 \leq j \leq |R_i|)$. Moreover, if t_{app} is the transition connecting *enabld* to *sem*, then $I(t_{app}) = [4, 4]$.

The definition of the communication part is the same as that of the proof of Theorem 1, we ignore omit repeating the construction. The main changes compared to the core model can be seen in Figure 8. □

6 Conclusions

In this paper, we have made a step forward in relating the membrane systems and time Petri nets. We connected membrane systems with promoters/inhibitors, membrane dissolution and priority for rules with time Petri nets by extending the Petri net model presented in [1]. We preserved the main characteristic of Petri nets, namely, the firings of the transitions can take place in any order: we do not impose any additional condition on the transition sequences in order to obtain a Petri net model equivalent to the general Turing machine. We can ignore the requirement of computing with maximal parallel transition sequences in the case of the Petri nets. Instead, our simulating Petri net model adopts the usual semantics: the fireable transitions can fire in any possible order.

References

1. Aman, B., Battyányi, P., Ciobanu, G., Vaszil, G.: Local time membrane systems and time Petri nets. Theoretical Computer Science (to appear) https://doi.org/10.1016/j.tcs.2018.06.013
2. Kleijn, J.H.C.M., Koutny, M., Rozenberg, G.: Towards a Petri Net Semantics for Membrane Systems. In: Freund R., Păun G., Rozenberg G., Salomaa A. (eds) Membrane Computing. WMC 2005. Lecture Notes in Computer Science, vol 3850. Springer, Berlin, Heidelberg (2006) 292–309
3. Merlin, P.M.: A Study of the Recoverability of Computing Systems. PhD Thesis, University of California, Irvine (1974)
4. Păun, G.: Membrane Computing. An Introduction. Springer, Berlin, Heidelberg (2002)
5. Păun, G., Rozenberg, G., Salomaa, A: The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
6. Petri, C.A.: Kommunikation mit Automaten. Dissertation, Universität Hamburg (1962)
7. Popova, L.: On Time Petri Nets. Journal of Information Processing and Cybernetics **27**(4) (1991) 227–244
8. Popova-Zeugmann, L.: Time and Petri Nets. Springer, Berlin, Heidelberg (2013)